



HACKATHON OMR CHALLENGE

TRIFORCE

Protótipo de Solução para Correção Automatizada de Provas e Simulados

Equipe:

7546 - Bianca Panacho Ferreira
8745 – Pedro Henrique Campos Moreira

Rio Paranaíba

11/2023

SUMÁRIO

1	VISÃO GERAL DA SOLUÇÃO.....	02
2	REQUISITOS.....	03
2.1	Especificações do sistema.....	03
2.2	Requisitos Funcionais.....	03
2.3	Requisitos Não Funcionais.....	04
3	ORGANIZAÇÃO DOS REQUISITOS.....	05
3.1	Casos de Uso.....	05
3.1.1	Diagrama de Casos de Uso.....	06
4	MODELO CONCEITUAL.....	06
5	INSTRUÇÕES DE USO DO EDUCADOR.....	07
5.1	Gabarito do Professor.....	07
5.2	Cartão Resposta do Aluno.....	07
6	METODOLOGIA USADA PARA DESLVLVER A SOLUÇÃO.....	07
6.1	Scrum.....	07
7	PROJETO DE INTERFACE.....	08
7.1	Sistema Web.....	08
7.2	Sistema Mobile.....	09
8	ESTRUTURAS DO CÓDIGO.....	10
9	DIAGRAMA DE ESTADOS DE NAVEGAÇÃO.....	17

1 Visão Geral da Solução

- O público alvo deste protótipo é composto por educadores, instituições de ensino, e professores, que desejam melhorar a eficiência na correção de provas e simulados. Ele pode ser usado por professores, educadores, administradores escolares, cujos interesses e expectativas foram considerados durante o desenvolvimento do sistema.
- O objetivo é fornecer uma solução que simplifique e automatize o processo de correção de provas e simulados, tornando-o mais rápido, preciso e eficiente. Ele visa melhorar a qualidade do ensino, economizando o tempo dos educadores e fornecendo feedback mais rápido aos alunos.
- O projeto contribuirá para a educação, permitindo que educadores dediquem mais tempo ao ensino e ao desenvolvimento de estratégias pedagógicas, em vez de gastar horas corrigindo manualmente provas e simulados. Além disso, o protótipo inclui relatórios individuais de cada aluno, mostrando quantas alternativas o aluno acertou, errou e a pontuação total.
- Existem várias soluções para a correção de provas, mas a maioria delas não oferece uma automação completa e personalizável. Há uma demanda significativa por uma ferramenta flexível que atenda às necessidades específicas de educadores e estudantes, de forma que não haja nenhum processo manual a ser feito.
- O escopo da solução contempla as seguintes funcionalidades:
 - Escaneamento de provas físicas;
 - Leitor ótico de marcas (OMR) para fazer uma visualização especial que consegue identificar marcas feitas a caneta preta nas posições de um formulário pré-definido (gabarito do professor e cartão resposta dos alunos);
 - Correção automática com base em chaves de resposta predefinidas (gabarito do professor);
 - Geração de relatórios individuais para cada aluno;
 - Análise estatística de desempenho por questão;
 - Suporte a questões de múltipla escolha;
 - Exportação de resultados em planilhas do Excel;
 - A Webcam, a câmera do celular ou a embutida no notebook são utilizadas para capturar imagens das provas ou simulados;
 - Interface Web, onde os usuários podem realizar login, cadastrar novos usuários, executar o programa de correção automática e visualizar os resultados.

- Evoluções futuras do sistema:
 - Correção de respostas dissertativas, verdadeiras ou falsas;
 - Integração com sistemas de gerenciamento de aprendizado (LMS).

2 Requisitos

2.1 Especificações do sistema

Especificações necessárias para o funcionamento efetivo do sistema e suas funcionalidades:

- Ter acesso a uma câmera;
- Possuir ambos os gabaritos: professor e aluno;
- É obrigatório marcar todas as alternativas no gabarito do professor e nos cartões de resposta dos alunos, não há o reconhecimento de alternativas em branco;
- Cadastrá-los nesta ordem, respectivamente.

2.2 Requisitos Funcionais

F1. Autenticação de Usuário:	Oculto ()
Descrição: O sistema deve permitir que os usuários façam login usando credenciais válidas para acessar as funcionalidades do sistema.	
F2. Cadastro de Usuário:	Oculto ()
Descrição: O sistema deve permitir que novos usuários se cadastrem fornecendo informações como nome, e-mail e senha.	
F3. Definição do Gabarito:	Oculto ()
Descrição: Os educadores devem poder definir o gabarito da prova, especificando as respostas corretas para cada pergunta.	
F4. Captura de Imagem	Oculto ()
Descrição: O sistema deve acessar a webcam do notebook, webcam ou a câmera do celular para capturar imagens do gabarito.	
F5. Correção Automatizada:	Oculto ()
Descrição: O sistema deve analisar as respostas dos alunos capturadas no imagem e compará-las com o gabarito, identificando respostas corretas e incorretas.	
F6. Geração de Relatório:	Oculto ()
Descrição: Após a correção automatizada, o sistema deve gerar um relatório contendo a pontuação total, número de acertos e erros de cada aluno.	

2.3 Requisitos Não Funcionais

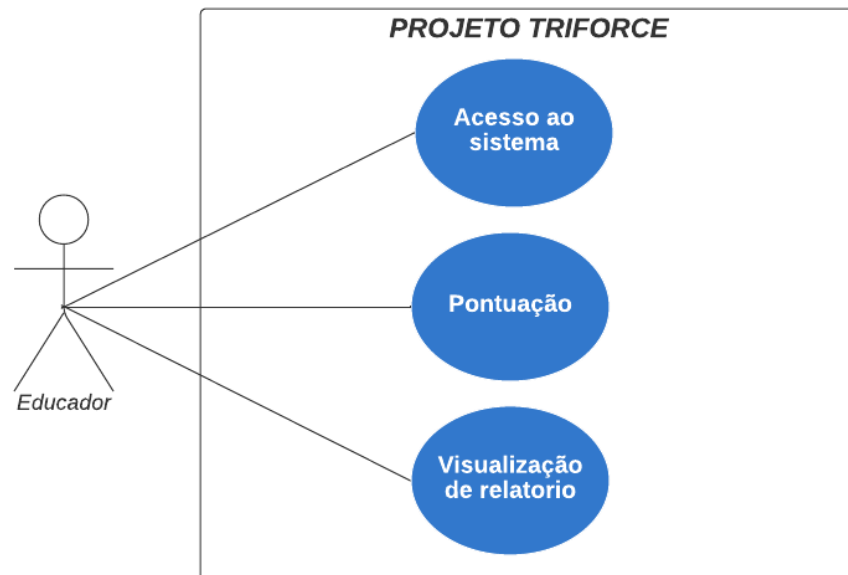
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Desejável	Permanente
Desempenho	O sistema deve ser responsivo e realizar a correção das provas em tempo real, mesmo com um grande número de alunos.	Desempenho	()	(X)
Segurança	As informações dos usuários e dos alunos devem ser armazenadas de forma segura.	Segurança	()	(X)
Usabilidade	A interface do usuário deve ser intuitiva e fácil de usar, permitindo que os educadores utilizem o sistema sem dificuldades.	Usabilidade	()	(X)
Confiabilidade	O sistema deve ser estável e confiável, minimizando falhas ou interrupções durante o processo de correção das provas.	Confiabilidade	()	(X)
Compatibilidade	O sistema deve ser compatível com diferentes dispositivos e navegadores para garantir uma experiência consistente para os usuários.	Compatibilidade	()	(X)
Manutenibilidade	O código do sistema deve ser modular e bem documentado, facilitando futuras atualizações e manutenções.	Manutenibilidade	()	(X)
Escalabilidade	O sistema deve ser projetado para ser escalável, permitindo que ele cresça e suporte um número crescente de usuários e provas sem comprometer o desempenho.	Confiabilidade	()	(X)

3. Organização dos Requisitos

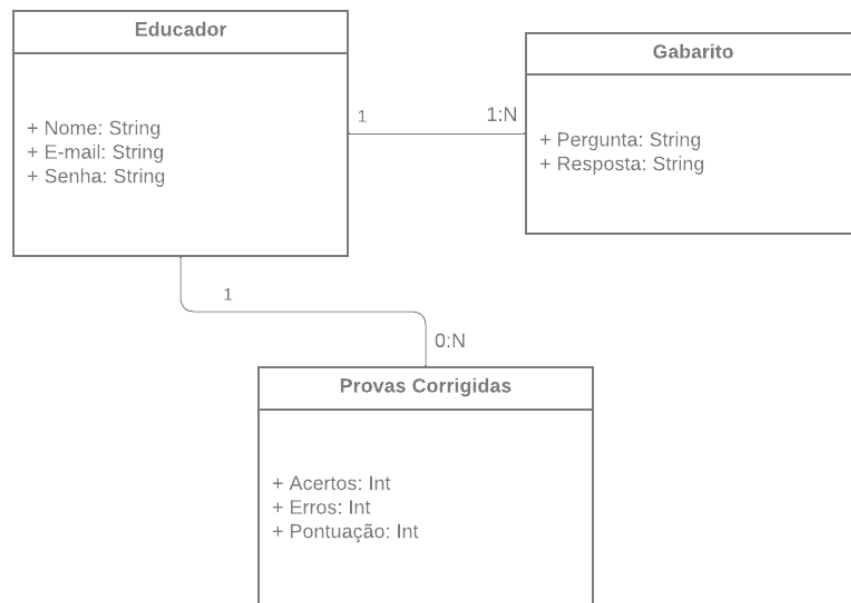
3.1 Casos de Uso

Nome	Atores	Descrição	Referências Cruzadas
Acesso ao sistema	Educador	Processo de autenticação do educador no sistema por meio de login e/ou cadastro.	F1 e F2
Pontuação	Educador	Permite que o educador cadastre o gabarito do professor e dos alunos e visualize o resultado da comparação entre ambos.	F3, F4 e F5
Visualização de Relatório	Sistema	O sistema gera um relatório de correção após a análise, contendo informações como pontuação total, número de acertos e erros de cada aluno.	F6

3.1.1 Diagrama de casos de uso



4. Modelo Conceitual do Domínio do Problema



5. Instruções de Uso para o Educador

5.1 Instruções Gabarito (Professor):

- Verifique a barra de tarefas, a janela "Gabarito" estará aberta;
- Posicione o gabarito em frente a câmera;
- Verifique se a grade das alternativas está azul;
- Pressione a tecla "s" para salvar o gabarito do professor;
- Pressione a tecla "q" para fechar a janela e abrir a do cartão-resposta.

5.2 Instruções Cartão de Respostas (Aluno):

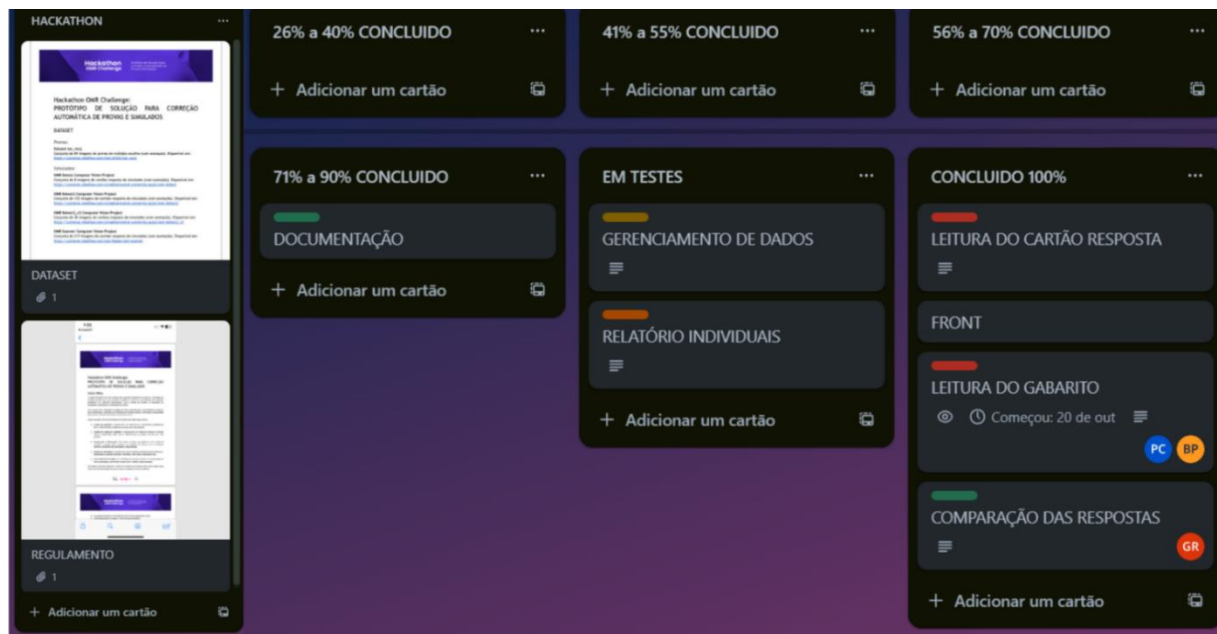
- A janela abrirá automaticamente;
- Pressione a tecla "p" para pausar e salvar o gabarito do aluno;
- Pressione "p" novamente para despausar e repetir o processo;
- Pressione a tecla "q" para fechar o sistema.

6. Metodologia Usada para Desenvolver a Solução

6.1 Scrum

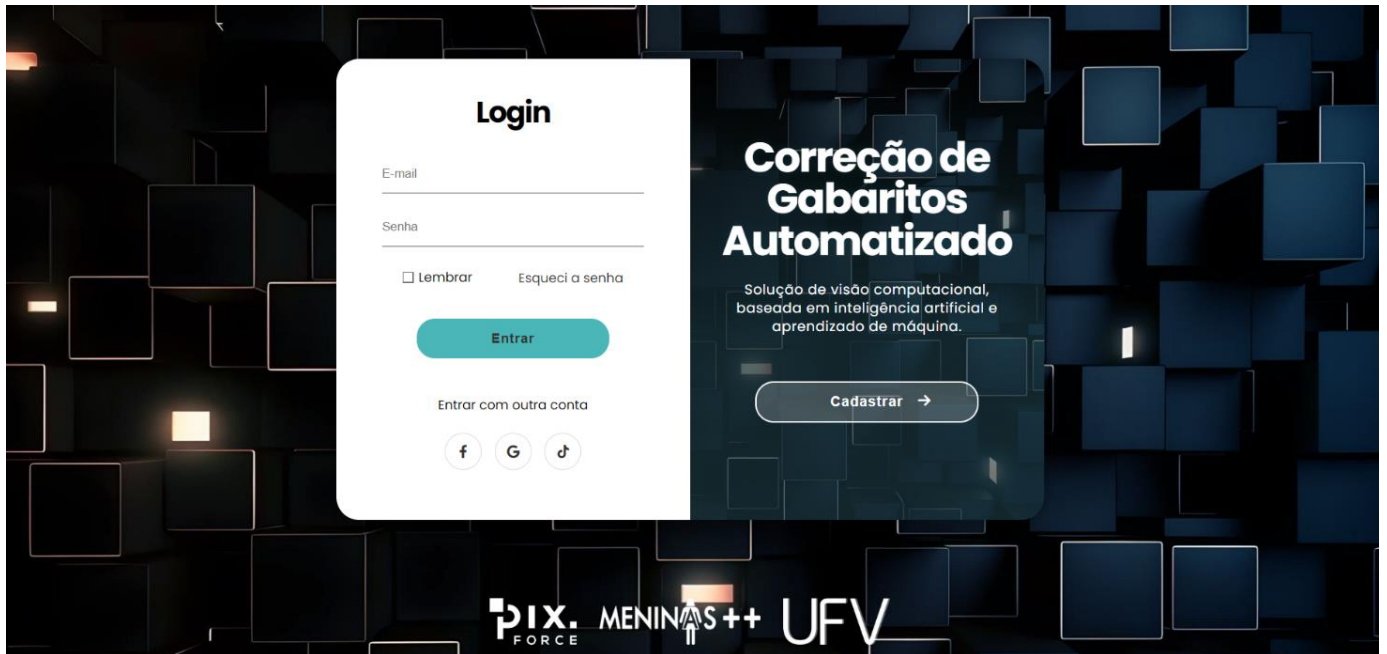
Para o desenvolvimento da solução utilizamos um processo baseado em Scrum, com reuniões diárias, semanais, e backlog para listagem dos requisitos. Utilizamos a ferramenta trello para controle do backlog:

1. **Daily:** 15 minutos diários, listagem dos backlogs feitos durante o dia e auxílio nas dúvidas, erros e tarefas incompletas.
2. **Sprint:** Semanalmente, reuniões presenciais para a atualização dos requisitos e backlogs.



7. Projeto de Interface

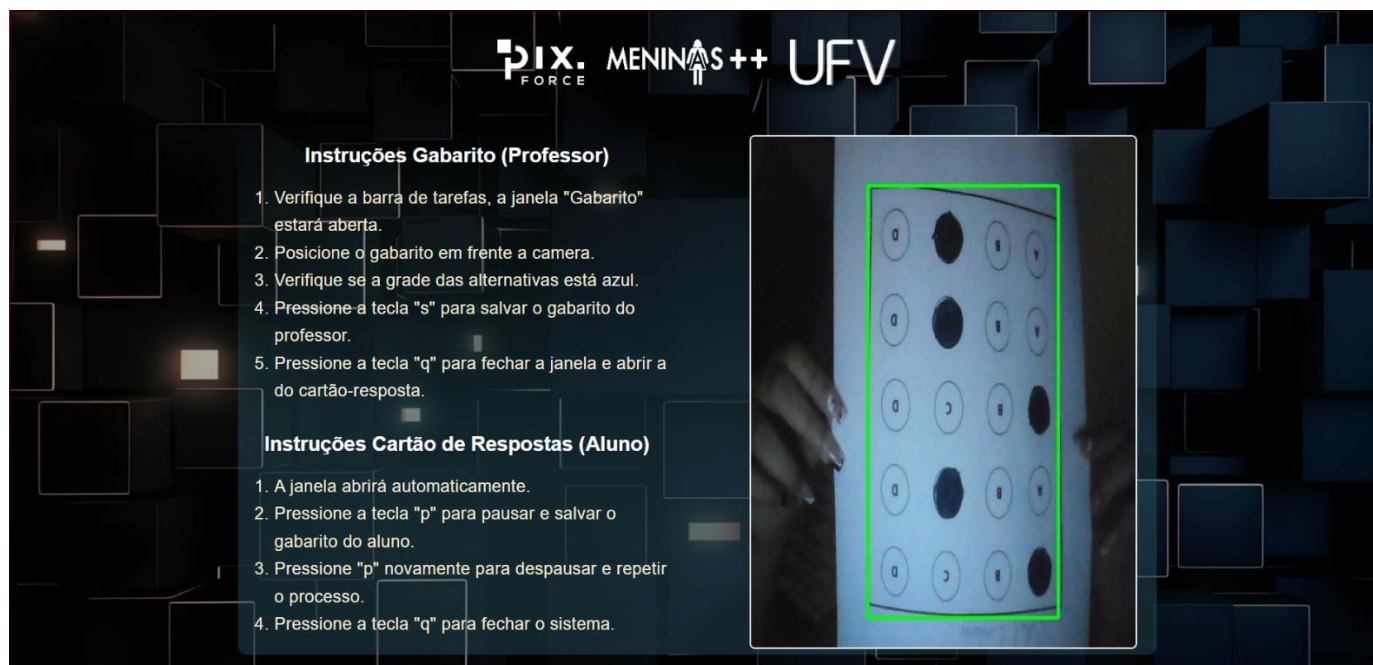
7.1 Sistema web



Tela de Login: Ocorre a autenticação do usuário no sistema, onde o usuário fornece credenciais (nome de usuário e senha) para acessar o sistema, e há um breve resumo a respeito das tecnologias utilizadas para a correção automatizada, como visão computacional.

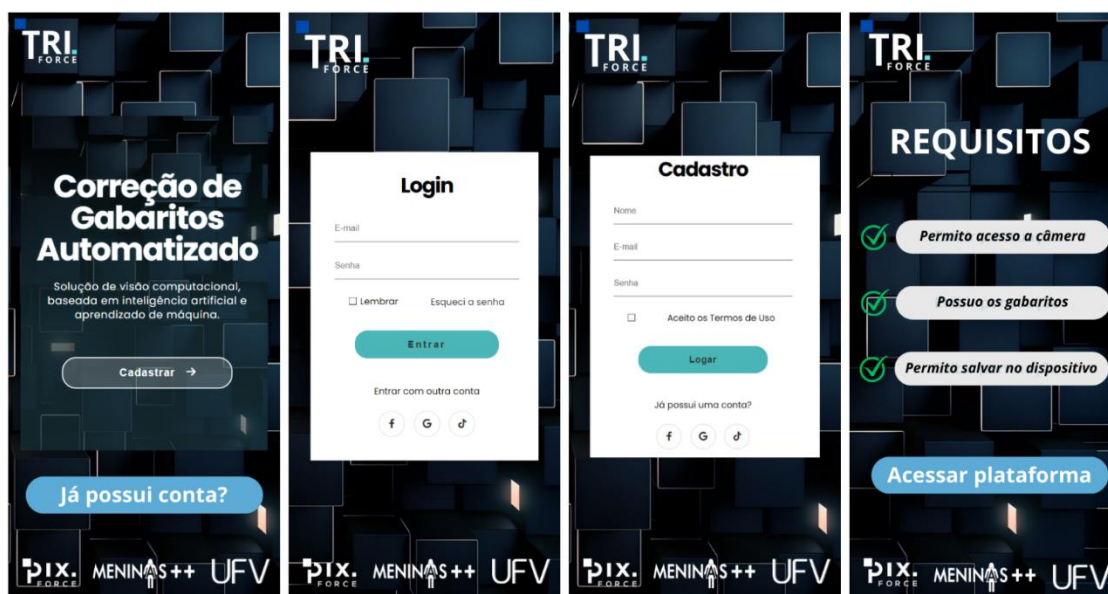


Tela de Requisitos: Lista todos os requisitos necessários para que o sistema seja funcional, e há um botão de direcionamento até a plataforma onde os gabaritos serão cadastrados.



Tela de Correção: Apresenta as orientações para o usuário e auxilia no entendimento da correção, trata-se de uma janela simples e fixa no navegador, que serve para exibir a pontuação do aluno mediante o gabarito do professor cadastrado.

7.2 Sistema Mobile





8. Estruturas do código

Metodologias desenvolvidas e aplicadas ao sistema web, explicando a lógica por trás do código, exemplificando o Front-End e Back-End.

- Visão geral do projeto:

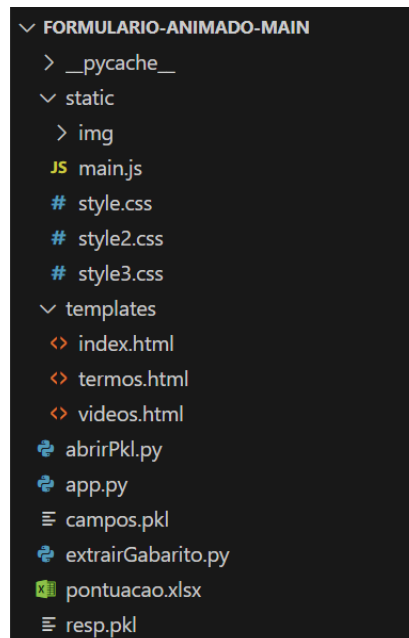
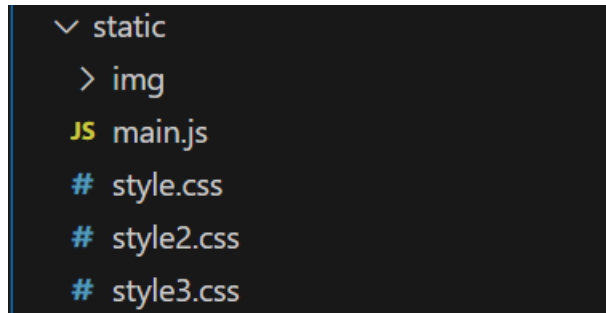


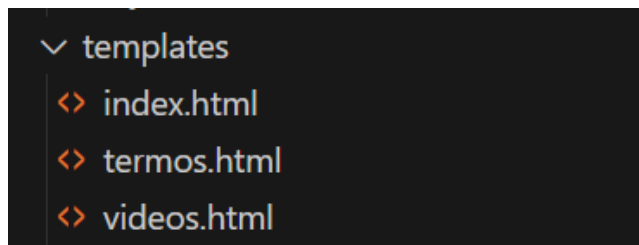
Imagem contendo todas as pastas e subpastas presentes no projeto, essa divisão foi feita baseada nas regras de implementação do **Flask**, micro framework para desenvolvimento web em Python.

Estruturação do código:

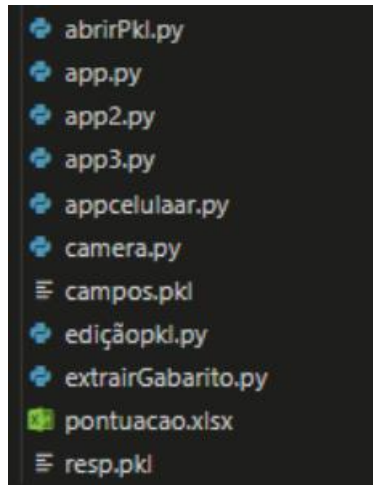
- **static** (contém todas as folhas de estilo, imagens e Javascript):
 - **img** (imagens utilizadas no projeto);
 - **main.js** (arquivo Javascript, servindo para estilização do CSS);
 - **style.css** (folha de estilo 1, linkada ao index.html);
 - **style2.css** (folha de estilo 2, linkada ao termos.html);
 - **style3.css** (folha de estilo 3, linkada ao videos.html).



- **Templates** (contém todas as páginas de Linguagem de Marcação de Hipertexto):
 - **index.html** (página de login, cadastro e aceitação dos termos de uso);
 - **termos.html** (página de requisitos e botão de acesso a plataforma);
 - **videos.html** (página de instruções de uso e janela que mostra a pontuação e acertos do aluno).



- **APP.PY**
 - **app.py** (arquivo Python que contém todo o código de OMR do projeto, comparação entre gabaritos e função de salvar no Excel).
 - **extrairGabarito.py** (responsável pela captura de imagens e envio ao arquivo campos.pkl).
 - **campos.pkl** (arquivo responsável pela mapeação do modelo de gabarito).
 - **resp.pkl** (arquivo responsável pela mapeação das respostas e auxílio na previsão).



- **Importação das bibliotecas**

```
import cv2
import pickle
import extrairGabarito as exG
import keyboard
import openpyxl
from flask import Flask, render_template, Response
```

OpenCV para manipulação de imagens e vídeos, é a biblioteca mais importante do código.

- **FLASK e variáveis globais**

```
app = Flask(__name__)

# PROFESSOR
# CADASTRAR GABARITO

camera = cv2.VideoCapture(0)

respostasCorretas = None
```

Inicialização do Flask

Inicialização da câmera e da variável de respostas corretas.

- Definição rotas de aplicação e armazenamento das respostas

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/termos')
def termos():
    return render_template('termos.html')

@app.route('/videos')
def videos():
    return render_template('videos.html')

def processar_video():
    global respostasCorretas
    pontuacao = 0

    video = cv2.VideoCapture(0)

    if not video.isOpened():
        print("Não foi possível abrir a webcam. Verifique se está conectada corretamente.")
        return

    campos = []
    resp = []
    with open('c:\\Users\\pedro\\Downloads\\formulario-animado-main (2)\\formulario-animado-main\\campos.pkl', 'rb') as arquivo_campos:
        campos = pickle.load(arquivo_campos)

    Para abrir o arquivo resp.pkl
    with open('c:\\Users\\pedro\\Downloads\\formulario-animado-main (2)\\formulario-animado-main\\resp.pkl', 'rb') as arquivo_respostas:
        resp = pickle.load(arquivo_respostas)

    pausado = False
    imprimir_respostas = False
    respostasCorretas = []
```

Definição das rotas de aplicação web com flask e função de encapsulamento para o 1 loop.

Leitura dos campos e respostas de arquivos pickle.

Lista para armazenar respostas corretas.

- Loop principal de captura de vídeo

```
# Loop principal de captura de vídeo
while True:
    _ , imagem = video.read() # Captura do frame da webcam

    # Verificação de erro na captura do frame
    if not _:
        print("Falha ao capturar frame da webcam.")
        break

    # Redimensionamento da imagem
    imagem = cv2.resize(imagem, (600, 700))

    # Extração do gabarito da imagem
    gabarito, bbox = exG.extrairMaiorCtn(imagem)

    1. gabarito is not None and bbox is not None:
    # Pré-processamento do gabarito
    imgGray = cv2.cvtColor(gabarito, cv2.COLOR_BGR2GRAY)
    ret, imgTh = cv2.threshold(imgGray, 70, 255, cv2.THRESH_BINARY_INV)
    cv2.rectangle(imagem, (bbox[0], bbox[1]), (bbox[0] + bbox[2], bbox[1] + bbox[3]), (255, 165, 0), 3)

    respostas = []

    # Iteração sobre os campos identificados
    for id, vg in enumerate(campos):
        x, y, w, h = int(vg[0]), int(vg[1]), int(vg[2]), int(vg[3])
        cv2.rectangle(gabarito, (x, y), (x + w, y + h), (0, 0, 255), 2)
        cv2.rectangle(imgTh, (x, y), (x + w, y + h), (255, 255, 255), 1)
        campo = imgTh[y:y + h, x:x + w]

        percentual_preto = (cv2.countNonZero(campo) / (h * w)) * 100

        if percentual_preto >= 10:
            cv2.rectangle(gabarito, (x, y), (x + w, y + h), (255, 0, 0), 2)
            respostas.append(resp[id])

    # Exibição das imagens e resultados
    cv2.imshow('Gabarito', gabarito)
```

Início do loop.

Extração do gabarito da imagem apresentada.

Iteração sobre os campos que foram identificados e Exibição da imagem.

```

_, buffer_imagem = cv2.imencode('.jpg', imagem)
frame_bytes_imagem = buffer_imagem.tobytes()

yield (b'--frame\r\n'
        b'Content-Type: image/jpeg\r\n\r\n'
        + frame_bytes_imagem + b'\r\n\r\n')

if keyboard.is_pressed('p'):
    pausado = not pausado
    print("Pausado" if pausado else "Despausado")

if keyboard.is_pressed('s') and not pausado and not imprimir_respostas:
    imprimir_respostas = True
    print("Respostas: ", respostas)
    respostasCorretas = respostas

if cv2.waitKey(1) & 0xFF == ord('q'):
    break

```

Conversão da imagem para formato JPEG e envio para a interface web.

Verificação da tecla 'p' para pausar/despausar.

Verificação da tecla 's' para imprimir respostas.

Verificação da tecla 'q' para pausar/despausar.

• Fim do loop principal e requisitos do segundo

```

video.release()
cv2.destroyAllWindows()

# Criação de um arquivo Excel para armazenar pontuações
workbook = openpyxl.Workbook()

fourcc = cv2.VideoWriter_fourcc('H264')
print(f"Suporta o codec H264? {cv2.VideoWriter_fourcc('H264') != 0}")

# Inicialização da variável de pausa
pausado = False

# Reabertura da câmera
video = cv2.VideoCapture(0)

# Verificação da abertura da câmera
if not video.isOpened():
    print("Não foi possível abrir a webcam. Verifique se está conectada corretamente.")
    exit()

# Leitura dos campos e respostas de arquivos pickle
campos = []
resp = []
with open('C:\\Users\\pedro\\Downloads\\formulario-animado-main (2)\\formulario-animado-main\\campos.pkl', 'rb') as arquivo_campos:
    campos = pickle.load(arquivo_campos)

with open('C:\\Users\\pedro\\Downloads\\formulario-animado-main (2)\\formulario-animado-main\\resp.pkl', 'rb') as arquivo_respostas:
    resp = pickle.load(arquivo_respostas)

# Criação do arquivo Excel
sheet = workbook.active

# Definição dos cabeçalhos na primeira linha da planilha
sheet['A1'] = 'Prova'
sheet['B1'] = 'Acertos'
sheet['C1'] = 'Erros'
sheet['D1'] = 'Pontuação'

# Inicialização da variável de prova fora do loop de captura
prova = 1

```

Finalização do primeiro loop.

Criação do Excel.

Configuração do codec de vídeo e inicialização da variável de pausa.

Definição do cabeçalho do Excel, seleção da planilha ativa no Excel e inicialização da variável prova.

• Loop 2

```
while True:
    key = cv2.waitKey(1) & 0xFF

    if key == ord('q'):
        break

    if not paused:
        imagem = video.read()
        imagem = cv2.resize(imagem, (600, 700))
        gabarito, bbox = ex0.extrairMaiorCtn(imagem)

        if gabarito is not None and bbox is not None:
            imgGray = cv2.cvtColor(gabarito, cv2.COLOR_BGR2GRAY)
            ret, imgTh = cv2.threshold(imgGray, 70, 255, cv2.THRESH_BINARY_INV)
            cv2.rectangle(imagem, (bbox[0], bbox[1]), (bbox[0] + bbox[2], bbox[1] + bbox[3]), (255, 165, 0), 3)

            respostas = []
            for id, vg in enumerate(campos):
                x, y, w, h = int(vg[0]), int(vg[1]), int(vg[2]), int(vg[3])
                cv2.rectangle(gabarito, (x, y), (x + w, y + h), (0, 0, 255), 2)
                cv2.rectangle(imgTh, (x, y), (x + w, y + h), (255, 255, 255), 1)
                campo = imgTh[y:y + h, x:x + w]

                percentual_preto = (cv2.countNonZero(campo) / (h * w)) * 100
                if percentual_preto >= 10:
                    cv2.rectangle(gabarito, (x, y), (x + w, y + h), (255, 0, 0), 2)
                    respostas.append(resp[id])

            erros = 0
            acertos = 0

            if len(respostas) == len(respostasCorretas):
                for num, res in enumerate(respostas):
                    if res == respostasCorretas[num]:
                        acertos += 1
                    else:
                        erros += 1
```

Inicialização do segundo loop e definição da tecla q para fechar as janelas do sistema.

Verifica se o vídeo não está pausado, o processamento ocorre apenas se o vídeo não estiver pausado.

Captura um frame do vídeo usando o método read() do objeto vídeo.

O resultado é uma tupla, o 1º elemento é um indicador de sucesso (True: leitura bem-sucedida) e o 2º elemento é a imagem.

A função está armazenando as respostas do aluno, a partir daí realiza o procedimento de correção.

Verifica se o número de respostas extraídas é igual ao número de respostas corretas.

Calcula a porcentagem de pixels pretos dentro de cada campo na imagem binária (imgTh) e, se for maior ou igual a 10%, adiciona a resposta correspondente à lista respostas: Erros = 0 e Acertos = 0. Inicializa contadores para o número de erros e acertos.

• Finalização do código

```
100 pontuacao = int(acertos * 3)
101 cv2.putText(imagem, 'Pontuacao:', (30, 140), cv2.FONT_HERSHEY_SIMPLEX, 1.2, (0, 0, 0), 3)
102 cv2.putText(imagem, 'Acertos:', (30, 180), cv2.FONT_HERSHEY_SIMPLEX, 1.2, (0, 0, 0), 3)
103 cv2.putText(imagem, 'Erros:', (30, 220), cv2.FONT_HERSHEY_SIMPLEX, 1.2, (0, 0, 0), 3)
104
105 cv2.putText(imagem, f'{pontuacao}', (250, 140), cv2.FONT_HERSHEY_SIMPLEX, 1.2, (255, 0, 0), 3)
106 cv2.putText(imagem, f'{acertos}', (100, 180), cv2.FONT_HERSHEY_SIMPLEX, 1.2, (0, 150, 0), 3)
107 cv2.putText(imagem, f'{erros}', (180, 220), cv2.FONT_HERSHEY_SIMPLEX, 1.2, (0, 0, 255), 3)
108
109 if paused:
110     cv2.putText(imagem, 'Video Pausado', (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
111
112 # Envio da imagem para a interface web
113 buffer_image = cv2.imencode('.jpg', imagem)
114 frame_bytes_image = buffer_image.tobytes()
115
116 yield (b'--frame\r\n'
117       b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes_image + b'\r\n\r\n')
118
119 if key != 255:
120     # Atualização da planilha Excel após pressionar qualquer tecla (exceto 'q')
121     prova += 1
122     sheet.cell(row=prova, column=1, value=prova - 1)
123     sheet.cell(row=prova, column=2, value=acertos)
124     sheet.cell(row=prova, column=3, value=erros)
125     sheet.cell(row=prova, column=4, value=pontuacao)
126
127 # Salvar o arquivo Excel
128 workbook.save("pontuacao.xlsx")
129
130 # Liberação da webcam e fechamento das janelas
131 video.release()
132 cv2.destroyAllWindows()
133
134 @app.route('/video_feed')
135 def video_feed():
136     return Response(processor_video(), mimetype='multipart/x-mixed-replace; boundary=frame')
137
138 # Execução da aplicação Flask
139 if __name__ == '__main__':
140     app.run(debug=True)
```

Definição das cores das letras que aparecem na tela do educador.

Envio da imagem para a interface web.

Atualização da planilha Excel após pressionar qualquer tecla (exceto 'q')

Salvamento do arquivo Excel, liberação da webcam e fechamento das janelas (fim do segundo loop).

Rota para o feed de vídeo e execução da aplicação Flask (final do código e do flask).

- **GADGETS**



Estamos demonstrando dois tipos de aplicações diferentes do sistema.

A primeira aplicação realiza a abertura da câmera através de uma câmera externa (WEBCAM) com variável sendo a igual dois, trocando essa variável por zero a câmera embutida do notebook é aberta.

A segunda trata-se de um link fornecido pelo aplicativo DroidCam, que o educador deve instalar em seu aparelho para utilizar a câmera do celular como forma de correção.

As 3 versões estão funcionais e disponíveis para escolha do educador antes de executar o programa.

- **Quantidade de questões e alternativas**



A parte grifada mostra como é possível atualizar a quantidade de questões e alternativas de cada prova e simulado, atendendo a solicitação do usuário.

É necessário adicionar as coordenadas das novas questões a grade ou a manta (janela Gabarito) que está no código, assim é possível corrigir qualquer gabarito desde que seja especificada a quantidade de questões e alternativas.

Os arquivos .pkl (pickle) são arquivos binários que realizam o mapeamento, criação e identificação das alternativas e questões. Para realizar modificações e implementações deve-se calcular as novas coordenadas a respeito das novas questões e alternativas, e posicioná-las entre os () apresentados no código acima.

9. Diagrama de Estados de Navegação

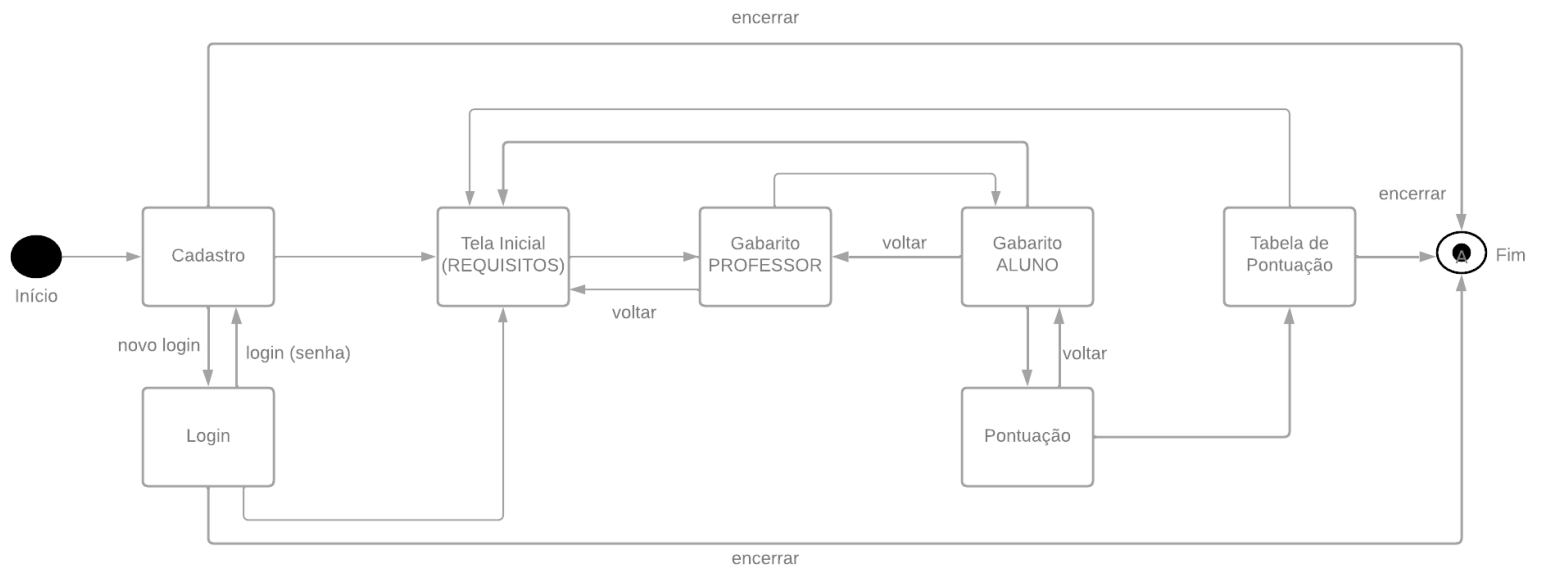


Diagrama usual para modelos web e mobile.